



PL/SQL Nested Blocks and SQL in PLSQL

- Jayendra Khatod

Objectives

- **Nested Blocks**
- **Variable Scope and Visibility**
- **Identifier Scope**
- **Write a successful SELECT statement in PL/SQL**
- **Declare the data type and size of a PL/SQL variable dynamically**
- **Write DML statements in PL/SQL**
- **Control transactions in PL/SQL**
- **Determine the outcome of SQL DML statements**

Nested Blocks and Variable Scope

- **Statements can be nested wherever an executable statement is allowed.**
- **A nested block becomes a statement.**
- **An exception section can contain nested blocks.**
- **The scope of an object is the region of the program that can refer to the object.**

Nested Blocks and Variable Scope

- **An identifier is visible in the regions in which you can reference the unqualified identifier:**
 - **A block can look up to the enclosing block.**
 - **A block cannot look down to enclosed blocks.**

Nested Blocks and Variable Scope

Example

```
•   ...  
•   x  BINARY_INTEGER;  
•   BEGIN  
•       ...  
•       DECLARE  
•           y  NUMBER;  
•       BEGIN  
•           ...  
•       END ;  
•       ...  
•   END ;
```

Scope of x

Scope of y

Identifier Scope

An identifier is visible in the regions where you can reference the identifier without having to qualify it:

- **A block can look up to the enclosing block.**
- **A block cannot look down to enclosed blocks.**

Qualify an Identifier

- **The qualifier can be the label of an enclosing block.**
- **Qualify an identifier by using the block label prefix.**

Qualify an Identifier

```
<<outer>>  
DECLARE  
    birthdate DATE;  
  
BEGIN  
    DECLARE  
        birthdate DATE;  
    BEGIN  
    ...  
    outer.birthdate :=  
    TO_DATE('03-AUG-1976','DD-MON-YYYY');  
    END;  
  
....  
END;
```


Determining Variable Scope

```
<<outer>>
DECLARE
    v_sal NUMBER(7,2) := 60000;
    v_comm NUMBER(7,2) := v_sal * 0.20;
    v_message VARCHAR2(255) := 'eligible for commission';
BEGIN
    DECLARE
        v_sal NUMBER(7,2) := 50000;
        v_comm NUMBER(7,2) := 0;
        v_total_comp NUMBER(7,2) := v_sal + v_comm;
    BEGIN
        v_message := 'CLERK not' || v_message;
        outer.v_comm := v_sal * 0.30;
    END;
    v_message := 'SALESMAN' || v_message;
END;
```

1 →

2 →

END;

Determining Variable Scope

Lets evaluate the PL/SQL block on the previous slide. Determine each of the following values according to the rules of scoping:

- 1. The value of V_MESSAGE in the sub-block.**
- 2. The value of V_TOTAL_COMP in the main block.**
- 3. The value of V_COMM in the sub-block.**
- 4. The value of V_COMM in the main block.**
- 5. The value of V_MESSAGE in the main block.**

SELECT Statements in PL/SQL

- Retrieve data from the database with **SELECT**.
- **Syntax**

```
SELECT  select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
WHERE   condition;
```

SELECT Statements in PL/SQL

- **The INTO clause is required.**
- **An Example**

```
DECLARE
    v_deptno NUMBER(4);
    v_location_id NUMBER(4);
BEGIN
    SELECT department_id, location_id
    INTO v_deptno, v_location_id
    FROM departments
    WHERE department_name = 'Sales';
    ...
END;
/
```

Retrieving Data in PL/SQL

- Retrieve the hire date and the salary for the specified employee.
- Example

```
DECLARE  
    v_hire_date employees.hire_date%TYPE;  
    v_salary employees.salary%TYPE;  
BEGIN  
    SELECT hire_date, salary  
    INTO v_hire_date, v_salary  
    FROM employees  
    WHERE employee_id = 100;  
    ...  
END;  
/
```

Retrieving Data in PL/SQL

- **Return the sum of the salaries for all employees in the specified department.**
- **Example**

```
DECLARE
    v_sum_sal NUMBER(10,2);
    v_deptno NUMBER NOT NULL := 60;
BEGIN
    SELECT SUM(salary) -- group function
    INTO v_sum_sal
    FROM employees
    WHERE department_id = v_deptno;
    DBMS_OUTPUT.PUT_LINE ('The sum salary is ' ||
    TO_CHAR(v_sum_sal));
END;
/
```


Inserting Data

- **Add new employee information to the EMP table.**
- **Example**

```
BEGIN
  INSERT INTO employees
    (employee_id, first_name, last_name, email,
     hire_date, job_id, salary)
  VALUES
    (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES',
     sysdate, 'AD_ASST', 4000);
END;
/
```

Updating Data

- **Increase the salary of all employees in the EMP table who are Analysts.**
- **Example**

```
DECLARE
    v_sal_increase employees.salary%TYPE := 800;
BEGIN
    UPDATE employees
    SET salary = salary + v_sal_increase
    WHERE job_id = 'ST_CLERK';

END;
```

Deleting Data

- **Delete rows that belong to department 10 from the EMP table.**
- **Example**

```
DECLARE
    v_deptno employees.department_id%TYPE := 10;
BEGIN
    DELETE FROM employees
    WHERE department_id = v_deptno;
    Commit;
END;
```

Naming Conventions

- **Use a naming convention to avoid ambiguity in the WHERE clause.**
- **Database columns and identifiers should have distinct names.**
- **Syntax errors can arise because PL/SQL checks the database first for a column in the table.**

Naming Conventions

DECLARE

hire_date employees.hire_date%TYPE;

sysdate hire_date%TYPE;

employee_id employees.employee_id%TYPE := 176;

BEGIN

SELECT hire_date, sysdate

INTO hire_date, sysdate

FROM employees

WHERE employee_id = employee_id;

END;

ERROR at line 1:

**ORA-01422: exact fetch returns more than
requested number of rows**

ORA-06512: at line 6

COMMIT and ROLLBACK Statements

- Use **COMMIT** and **ROLLBACK SQL** statements to terminate a transaction explicitly.

SQL Cursor

- **A cursor is a private SQL work area.**
- **There are two types of cursors:**
 - **Implicit cursors**
 - **Explicit cursors**
- **The Oracle Server uses implicit cursors to parse and execute your SQL statements.**
- **Explicit cursors are explicitly declared by the programmer.**

SQL Cursor Attributes

- Using SQL cursor attributes, you can test the outcome of your SQL

SQL%ROWCOUNT	Number of rows affected by the most recent SQL statement (an integer value)
SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement affects one or more rows
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement does not affect any rows
SQL%ISOPEN	Always evaluates to FALSE because PL/SQL closes implicit cursors immediately after they are executed

SQL Cursor Attributes

- **Delete rows that have the specified employee ID from the copy_emp table. Print the number of rows deleted.**
- **Example**

```
DECLARE  
    v_employee_id copy_emp.employee_id%TYPE := 176;  
BEGIN  
    DELETE FROM copy_emp  
    WHERE employee_id = v_employee_id;  
    DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' row  
    deleted.');
```

END;

Summary

- **PL/SQL block structure: Nesting blocks and scoping rules**
- **Use SQL in the PL/SQL block:
SELECT, INSERT, UPDATE, DELETE
COMMIT, ROLLBACK, SAVEPOINT**

Summary

- **There are two cursor types: implicit and explicit.**
- **Implicit cursor attributes verify the outcome of DML statements:**
 - **SQL%ROWCOUNT**
 - **SQL%FOUND**
 - **SQL%NOTFOUND**
 - **SQL%ISOPEN**
- **Explicit cursors are defined by the programmer.**



Thank You !